УДК 004.75

Ю. А. Шичкина, М. Х. А. Аль-Марди, М. С. Куприянов
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Оптимизация параллельного алгоритма путем сокращения времени на межпроцессорный обмен данными

Приводятся результаты исследований в области построения эффективных по времени и объему вычислительных ресурсов параллельных алгоритмов для вычислительных систем с распределенной памятью. Представлен метод, который позволяет усовершенствовать расписание выполнения параллельного алгоритма по времени выполнения за счет перераспределения операций между процессами. В итоге уменьшается объем передаваемых сообщений между процессами и сокращается время, затрачиваемое на межпроцессорный обмен данными. Объем вычислительных ресурсов при этом не меняется. Метод основан на списках смежности, соответствующих информационному графу алгоритма. Метод может быть применен в совокупности с другими методами построения расписания выполнения алгоритмов, например ориентированных на оптимизацию по числу вычислительных узлов, для достижения оптимального расписания по ряду параметров.

Параллельный алгоритм, оптимизация алгоритма, информационный граф, время выполнения операции, расписание выполнения алгоритма, процесс, процессор, межпроцессорная передача данных

На протяжении нескольких десятилетий, включая настоящее время, в области параллельных технологий наблюдается тенденция отставания развития математического аппарата, поддерживающего параллельные вычисления, от более высокого развития архитектуры вычислительных систем. В связи с этим на прекрасных по мощностям вычислительных системах часто так и не удается достичь теоретически возможного ускорения решения прикладных задач. Другими словами, современные высокопроизводительные системы используются неэффективно, а многие задачи решаются с большой временной задержкой.

При этом надо отметить, что начиная с 1980-х гг. в области теории и практики параллельных вычислений проводились исследования. Все их можно разбить на следующие категории:

– разработка параллельных вычислительных систем;

– анализ эффективности параллельных вычислений для оценки получаемого ускорения вычислений и степени использования всех возможностей компьютерного оборудования при параллельных способах решения задач;

– формирование общих принципов разработки параллельных алгоритмов для решения сложных вычислительно трудоемких задач;

– создание и развитие системного программного обеспечения для параллельных вычислительных систем (например, MPI);

– создание и развитие параллельных алгоритмов для решения прикладных задач в разных областях практических приложений.

Больше всего исследовательских работ проводилось ранее и проводится теперь в последней области. Здесь можно выделить работы, связанные, например, с моделированием расчета электромагнитного излучения на основе многоядерных процессоров и кластеров параллельной архитектуры GPU с использованием гибридного параллельного алгоритма MPI-OpenMP и MPI-CUDA [1]. Проводятся исследования в области трансляции больших объемов данных в IoT (Internet of Things) с применением высокопроизводительных технологий [2]. Много исследований параллельных алгоритмов проводится в области дифференциальных уравнений [3] и систем линейных уравнений [4].

Начало анализа масштабируемости параллельных программ и компьютерных систем относится к

1967 г., когда сотрудник IBM Джин Амдал (Gene Amdahl) опубликовал статью [5], которая позже стала классической. Дальнейшая оценка ускорения вычислительных алгоритмов, использования ресурсов и производительности вычислительных систем была получена во многих других исследованиях [6].

Новейшие суперкомпьютеры имеют очень большое число вычислительных узлов/ядер, но многие практические приложения не могут достичь максимальной производительности на этих суперкомпьютерах из-за недостаточно развитого аппарата исследования параллелизма в приложениях. В области формирования общих принципов разработки параллельных алгоритмов можно выделить исследования по планированию выполнения операций алгоритма [7]. Одним из недостатков этих алгоритмов является их NP-полнота.

Проблема построения расписания параллельного алгоритма с оптимальным временем выполнения была освещена в различных исследованиях с 1980-х гг. С тех пор были созданы различные алгоритмы построения расписания, основанные на методах нахождения максимального потока в транспортной сети, на основе ветвей и границ, динамическом программировании, а также итерационные (например, генетические алгоритмы, алгоритм отжига) [8]. Большинство из этих алгоритмов, в частности основанные на нахождении максимального потока в транспортной сети, имеют псевдополиномиальную сложность.

Таким образом, процессы распараллеливания алгоритмов по-прежнему проводятся «вручную» и ложатся на плечи разработчика. Это одна проблема.

Другая проблема заключается в том, что при распараллеливании алгоритмов практически никогда не учитывается передача данных между вычислительными узлами. Известно, что график зависимости скорости вычислений от числа вычислительных узлов при реализации алгоритма на системах с распределенной памятью имеет форму нормального распределения. Существует такое число вычислительных узлов, при котором скорость работы параллельного алгоритма будет наивысшей. При превышении этого числа скорость начнет падать за счет увеличения обмена данными между вычислительными узлами. Но если обмен данными между узлами уменьшить, то можно значительно поднять скорость работы алгоритма даже при имеющемся объеме вычислительных ресурсов. В данной статье авторами приведены результаты исследований по проблеме оптимизации параллельных алгоритмов по объему межпроцессорных передач данных для случая, когда все операции алгоритма различаются по времени выполнения. Случай, когда время выполнения всех операций алгоритма одинаково рассмотрен в статьях [9], [10].

Описание задачи и условий ее реализации. Пусть для оптимизируемого алгоритма и вычислительной системы заданы следующие ограничения:

– вычислительная система не ограничена по количеству процессоров (ядер);

– каждая операция обладает одинаковым объемом входных данных;

– время выполнения каждой операции разное;

– время передачи данных между двумя любыми процессорами постоянное, условно равное 1 ед.

Будем также полагать, что для преобразуемого алгоритма произвольным способом построен информационный граф в некоторой параллельной форме. Особенностью предлагаемых далее методов является то, что объем вычислительных ресурсов оптимизированного алгоритма будет не больше объема вычислительных ресурсов исходного алгоритма. Поэтому перед применением этих методов желательно провести оптимизацию алгоритма по ширине или настройку алгоритма под определенный объем вычислительных ресурсов.

Суть метода оптимизации алгоритма по объему межпроцессорных передач с учетом времени выполнения каждой операции заключается в перестановке вершин внутри яруса и между ярусами таким образом, чтобы минимизировать общее время выполнения алгоритма.

Условие разного времени выполнения операций, в несколько раз усложняет методы построения расписания алгоритма, так как необходимо отслеживать не только прямые, но и транзитивные информационные связи между операциями алгоритма. Но, с другой стороны, это условие позволяет сократить общее время работы алгоритма за счет не только ликвидации пузырей, но и более плотного распределения операций.

Будем в дальнейшем полагать под ранним сроком выполнения операции минимальное время окончания выполнения операции с момента начала выполнения всего алгоритма. Поздний срок выполнения операции – максимально допустимое время окончания выполнения операции с момента начала выполнения всего алгоритма и с учетом ограничения на время выполнения алгоритма.

Тогда при проходе по ярусам информационного графа в направлении от первого к последнему будут получены ранние сроки выполнения операций и самого алгоритма, а при переборе вершин по ярусам в направлении с конца к началу будут получены поздние сроки. Поэтому выбор направления перебора вершин зависит не только от качества работы метода в данном направлении, но и от конечной цели исследования алгоритма.

Будем называть две вершины А и В *строго бинарно связанными*, если они связаны одним ребром и полустепень исхода вершины А равна полустепени захода вершины В и равна 1: $d^+(\text{A}) = d^-(\text{B}) = 1$.

Будем называть две вершины А и В *бинарно связанными*, если они связаны одним ребром и полустепень исхода вершины А больше 1, а полустепень захода вершины В равна 1: $d^+(\text{A}) > 1$, $d(\text{B}) = 1$.

Метод построения расписания в направлении перебора вершин по ярусам с начала в конец. Пусть m – общее число групп, полученное формализованным методом, например методом оптимизации информационного графа по ширине с помощью матрицы или списка смежности [9], k – номер текущей группы, nk – число вершин в группе с номером k, i – номер текущей вершины в группе с номером k, j – номер текущей вершины в группе с номером $k-1$.

1. Процесс перестановки вершин начинается с первой группы. Номер текущей группы $k = 1$. Все процессоры в группе полагаем не отмеченными.

2. Для M_k найти множество ранних сроков выполнения операций из группы $M_{(k+1)}$ с учетом передачи данных по правилам (1, 2).

Правило 1. Пусть $M_{(k+1)} = \{v_1, \ldots, v_j, \ldots, vn_{(k+1)}\}$ – группа, в которой происходит перестановка вершин, $T_{(k+1)} = \{t_1, \ldots, t_j, \ldots, tn_{(k+1)}\}$ – непосредственное время выполнения каждой операции из группы $M_{(k+1)}$.

Тогда время выполнения каждой операции из группы $M_{(k+1)}$ с учетом передачи данных на i-м процессоре

$$T' = \left\{T'_{(k+1)}\right\}_i = \left\{t_j + \alpha\right\},\ j = 1\ldots n_{(k+1)},\ i = 1\ldots n_k,$$
$$\alpha = \begin{cases} 1, \text{ если } \exists\left(M_{ki}, M_{(k+1)j}\right), \\ 0, \text{ если } \nexists\left(M_{ki}, M_{(k+1)j}\right). \end{cases} \quad (1)$$

Правило 2. Пусть $M_{(k+1)} = \{v_1, \ldots, v_j, \ldots, vn_{(k+1)}\}$ – группа, в которой происходит перестановка вершин, $M_k = \{u_1, \ldots, u_j, \ldots, un_k\}$ – группа вершин на закрепленном (фиксированном) ярусе, $T' = \left\{T'_{(k+1)}\right\}_i\, i\,(i = 1\ldots nk)$ – время выполнения каждой операции из группы $M_{(k+1)}$ с учетом передачи данных на i-й процессор.

Тогда ранний срок выполнения каждой операции из группы $M_{(k+1)}$ с учетом передачи данных на i-м процессоре

$$\tau' = \left\{\tau'_{k+1}\right\}_i = \left\{T'_{(k+1)i} + \max\left(\tau_i, \tau_r\right)\right\},$$
$$\text{для } \forall \text{r: } \exists\left(M_{kr}, M_{(k+1)j}\right), \quad (2)$$
$$r = 1\ldots n_k,\ j = 1\ldots n_{k+1}$$

1. Найти оптимальное расписание выполнения операций группы M_{k+1}. Оптимальным будем полгать такое расписание, по которому выполнение всех операций группы M_{k+1} заканчивается раньше, чем в других расписаниях. Для поиска оптимального расписания необходимо:

а) составить множество P всех возможных расписаний выполнения операций группы M_{k+1} после операций группы M_k;

б) в каждом расписании P_r, $r = 1\ldots n_p$ найти максимальный ранний срок выполнения операции:

$$P_{r\max} = \max\left(\tau'_{rj}\right),\ r = 1\ldots n_p,\ j = 1\ldots n; \quad (3)$$

в) среди всех $P_{r\max}$ найти минимальное значение. Расписание, которому будет соответствовать это значение, будет оптимальным расписанием.

Формализованно это можно описать так:

Правило 3.

$$P_{\min} = P_s\colon P_{s\max} = \min\left[\max\left(\tau'_{rj}\right)\right],\ r = 1\ldots n_p,$$
$$j = 1\ldots n,\ n_p = n(n!),\ n = \max\ \max\left(n_k, n_{(k+1)}\right). \quad (4)$$

Расписаний, удовлетворяющих условию (4), может быть несколько. В этом случае из них необходимо выбрать то расписание, суммарное значение ранних сроков выполнения которого наименьшее.

Формализованно это можно описать так:

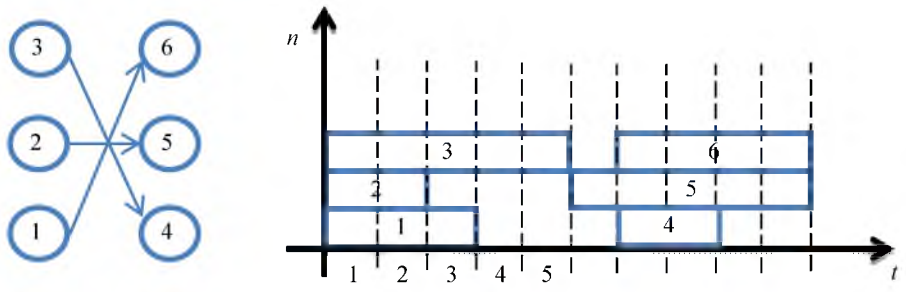
Правило 4.

$$P_{\min} = P_s\colon P_{s\max} = \min\left[\max\left(\tau'_{rj}\right)\right],$$
$$\sum_{k=1}^{n} \tau'_{sj} = \min_s\left(\sum_{j=1}^{n} \tau'_{rj}\right), \quad (5)$$
$$r = 1\ldots n_p,\ j = 1\ldots n,\ n_p = n(n!),\ n = \max\left(n_k, n_{k+1}\right).$$

2. Переставить вершины в соответствии с найденным расписанием.

3. Если $k < m$, то $k = k + 1$ и шаг 2. Иначе шаг 6.

4. Конец метода.



Примеры перестановки операций в алгоритме.

Пример применения правила 1. Пусть при распределении вершин по группам получились следующие фрагмент графа и расписание.

Для $i = 2$ получаем время выполнения операций:

$$\{t_4 + 1, t_5, t_6 + 1\} = \{3, 5, 5\}.$$

Ко времени четвертой и шестой операций будет прибавлена единица, так как если их переставить на второй процессор, то они будут вынуждены ожидать информацию с других процессоров.

Для M_2 {4, 5, 6} время выполнения операций с учетом передачи данных T' с группы M_1 {1, 2, 3} по (1) будет равно: $T' = \{\{3, 6, 4\}, \{3, 5, 5\}, \{2, 6, 5\}\}$.

Пример применения правила 2. Пусть при распределении вершин по группам получились фрагмент графа и расписание, как на рисунке.

По правилу 1 для группы M_2 будет получено время выполнения операций: $T' = \{\{3, 6, 4\}, \{3, 5, 5\}, \{2, 6, 5\}\}$.

По правилу 2, например, для $i = 2$ будет получено множество ранних сроков выполнения операций из группы M_2: $\tau'_2 = \{3 + \max(2; 5), 5 + \max(2), 5 + \max(2; 3)\} = \{8, 7, 8\}$.

Для M_1 {1, 2, 3} множество ранних сроков выполнения операций группы M_2 {4, 5, 6} с учетом передачи данных по формуле (2) будет равно:

$$\tau' = \begin{Bmatrix} \{3 + \max(3, 5), 6 + \max(3, 2), 4 + \max(3)\} \\ \{3 + \max(2, 5), 5 + \max(2), 5 + \max(2, 3)\} \\ \{2 + \max(5), 6 + \max(5, 2), 5 + \max(5, 3)\} \end{Bmatrix},$$

$$\tau' = \{\{8, 8, 7\}, \{8, 7, 8\}, \{7, 11, 10\}\}.$$

Пример множества P всех возможных расписаний выполнения операций группы M_{k+1} после операций группы M_k.

Пусть

$$\tau' = \begin{Bmatrix} \{\tau_{11}, \tau_{12}, \tau_{13}\} \\ \{\tau_{21}, \tau_{22}, \tau_{23}\} \\ \{\tau_{31}, \tau_{32}, \tau_{33}\} \end{Bmatrix}.$$

Тогда

$$P = \begin{Bmatrix} \{\tau_{11}, \tau_{22}, \tau_{33}\}, \{\tau_{11}, \tau_{23}, \tau_{32}\}, \{\tau_{12}, \tau_{21}, \tau_{33}\}, \\ \{\tau_{12}, \tau_{23}, \tau_{31}\}, \{\tau_{31}, \tau_{22}, \tau_{13}\}, \{\tau_{31}, \tau_{23}, \tau_{12}\} \\ \{\tau_{21}, \tau_{12}, \tau_{33}\}, \{\tau_{21}, \tau_{13}, \tau_{32}\}, \{\tau_{22}, \tau_{13}, \tau_{31}\}, \\ \{\tau_{22}, \tau_{31}, \tau_{13}\}, \{\tau_{23}, \tau_{12}, \tau_{31}\}, \{\tau_{23}, \tau_{31}, \tau_{12}\} \\ \{\tau_{31}, \tau_{12}, \tau_{23}\}, \{\tau_{31}, \tau_{12}, \tau_{33}\}, \{\tau_{32}, \tau_{13}, \tau_{21}\}, \\ \{\tau_{32}, \tau_{31}, \tau_{12}\}, \{\tau_{33}, \tau_{11}, \tau_{22}\}, \{\tau_{33}, \tau_{12}, \tau_{21}\} \end{Bmatrix}.$$

Всего P будет состоять из $n_p = n(n!)$ сочетаний, где $n = \max(n_k, n_{k+1})$.

Метод ускоренного перебора расписаний.

Обозначения. Пусть R – номер процессора, с которого начинается составление расписания; r – номер текущего процессора; j – номер операции в группе $M_{(k+1)}$; P – множество всех выбранных расписаний; s – номер текущего расписания; p_{sr} – операция, выполняемая на r-м процессоре по расписанию с номером s; τ'_{sr} – ранний срок выполнения p_{sr}-й операции на r-м процессоре по расписанию с номером s.

1. Пусть $R = 1$, $s = 1$.

2. Пусть $r = 1$, $p_{sr} = 0$ для всех $r = 1 \ldots n_{k+1}$.

3. Найти минимальный ранний срок во множестве τ'_r. $\tau'_{rj} = \min(\tau'_{rl})$, $l = 1 \ldots n_{(k+1)} / \{p_{sr}\}$, $r = 1 \ldots n_{(k+1)}$. Положить $p_{sr} = j$, $\tau'_{sr} = \tau'_{rj}$.

4. Если $r < n_k$, то положить $r = r + 1$ и шаг 3. Если на шаге 3 было найдено несколько минимальных ранних сроков, то шаг 4 выполняется для каждого из них. Если $r = n_k$, то шаг 5.

5. Если $R < n_k$, то положить $R = R + 1$ и шаг 2. Иначе – шаг 6.

6. Конец алгоритма.

В результате трудоемкость всего алгоритма по поиску возможный расписаний составит $O(n_k n_{(k+1)}) = O(n^2)$, где $n = \max(n_k, n_{(k+1)})$.

Пример ускоренного перебора расписаний. Пусть группа $M_{(k+1)} = \{3, 7, 9\}$. Для этой группы в соответствии с (1), (2) найдено множество τ':

$$\tau' = \begin{Bmatrix} \{8, 8, 7\} \\ \{7, 7, 8\} \\ \{7, 11, 10\} \end{Bmatrix}.$$

Следуя вышеизложенному методу:

1. В первом множестве найдем минимальный ранний срок и номер его операции: $p_{11} = 3$ (третий столбец – третий порядковый номер операции), $\tau'_{11} = 7$.

2. Во втором множестве среди всех ранних сроков за исключением 3-го (так как $p_{11} = 3$) найдем минимальный ранний срок: $p_{12} = 1$ (первый столбец – первый порядковый номер операции), $\tau'_{12} = 7$.

3. В третьем множестве среди всех ранних сроков за исключением 1-го, 3-го (так как $p_{11} = 3$, $p_{12} = 1$) найдем минимальный ранний срок: $p_{13} = 2$ (второй столбец – второй порядковый номер операции), $\tau'_{13} = 11$.

Получили первое расписание: p_1 {операция 3, операция 1, операция 2}. Возьмем номера операций из $M_{k+1} = \{3, 7, 9\}$, получим расписание:

p_1 {9, 3, 7} с ранними сроками выполнения

$$\tau'_1 = \{7, 7, 11\}.$$

4. Теперь положим $p_{21} = p_{11}$ и вернемся к шагу 2. Во втором множестве среди всех ранних сроков за исключением 3-го (так как $p_{21} = p_{11} = 3$) есть второй минимальный ранний срок: $p_{22} = 2$ (второй столбец – второй порядковый номер операции), $\tau'_{22} = 7$.

5. В третьем множестве среди всех ранних сроков за исключением 2-го, 3-го (так как $p_{21} = 3$, $p_{22} = 2$) найдем минимальный ранний срок: $p_{23} = 1$ (первый столбец – первый порядковый номер операции), $\tau'_{13} = 7$.

Получили второе расписание: p_2 {операция 3, операция 2, операция 1}. Возьмем номера операций из $M_{k+1} = \{3, 7, 9\}$, получим расписание:

p_2 {9, 7, 3} с ранними сроками выполнения

$$\tau'_2 = \{7, 7, 7\}.$$

6. Теперь начнем перебор множеств со второй группы и будем составлять третье расписание. Во втором множестве среди всех ранних сроков найдем минимальный ранний срок: $p_{32} = 1$ (первый столбец – первый порядковый номер операции), $\tau'_{32} = 7$.

7. В первом множестве среди всех ранних сроков за исключением 1-го (так как $p_{32} = 1$) найдем минимальный ранний срок и номер его операции: $p_{31} = 3$ (третий столбец – третий порядковый номер операции), $\tau'_{31} = 7$.

8. В третьем множестве среди всех ранних сроков за исключением 1-го, 3-го (так как $p_{32} = 1$, $p_{31} = 3$) найдем минимальный ранний срок: $p_{33} = 2$ (второй столбец – второй порядковый номер операции), $\tau'_{33} = 11$.

Получили второе расписание: p_2 {операция 3, операция 2, операция 1}. Возьмем номера операций из $M_{k+1} = \{3, 7, 9\}$, получим расписание:

p_3 {9, 3, 7} с ранними сроками выполнения

$$\tau'_3 = \{7, 7, 11\}.$$

Продолжая таким образом, получим в итоге следующую совокупность расписаний:

Расписание	Ранние сроки
p_1 {9, 3, 7}	$\tau'_1 = \{7, 7, 11\}$
p_2 {9, 7, 3}	$\tau'_2 = \{7, 7, 7\}$

Все остальные расписания будут их копией. Всего будет найдено 9 расписаний. При полном переборе $n(n!) = 3(3!) = 18$. Таким образом удалось в два раза меньше провести операций.

Следуя далее методу поиска оптимального расписания для группы M_{k+1} найдем в каждом из найденных расписаний максимальный ранний срок (3): $P_{r\max} = \{11, 7\}$ (11 для первого расписания p_1, 7 – для второго p_2).

Далее найдем среди значений множества $P_{r\max}$ минимальный элемент (4): $P_{\min} = 7$. Это значение единственное и соответствует расписанию p_2 {9, 7, 3}.

В результате оптимальным расписанием для группы M_{k+1} будет p_2 {9, 7, 3}.

Описанный в статье метод позволяет осуществлять перестановку операций между вычислительными узлами так, чтобы максимально сократить время, затрачиваемое на передачу дан-

ных. При этом в методе учитывается время выполнения самих операций и характер связи операций между собой. Так, при наличии нескольких операций, зависящих по данным от предыдущей операции, на вычислительных узлах операции будут расставлены так, чтобы суммарное время на их выполнение с учетом пересылки данных было минимальным.

Алгоритм, построенный по данному методу, имеет небольшую трудоемкость $O(n^2m)$, где n – число вычислительных узлов; m – число ярусов информационного графа. Метод основан на списках смежности, что позволяет значительно экономить память по сравнению с матрицей смежности.

Метод оптимизации параллельного алгоритма по времени с учетом межпроцессорных передач может быть успешно применен в случаях построения расписания выполнения задач в распределенных вычислительных системах.

Публикация выполнена в рамках государственной работы «Инициативные научные проекты» базовой части государственного задания Министерства образования и науки Российской Федерации (Задание № 2.6553.2017/8.9) и в рамках договора № 02.G25.31.0149 от 01.12.2015.

СПИСОК ЛИТЕРАТУРЫ

1. Parallel Numerical Simulations of Three-Dimensional Electromagnetic Radiation with MPI-CUDA Paradigms / Bing He, Long Tang, Jiang Xie, XiaoWei Wang, AnPing Song // Math. Probl. in Engin. Vol. 2015. Art. ID 823426. 9 p.

2. Qin Jiancheng, Lu Yiqin, Zhong Yu. Parallel Algorithm for Wireless Data Compression and Encryption // J. of Sensors. Vol. 2017. Art. ID 4209397. 11 p.

3. A Parallel Algorithm for the Two-Dimensional Time Fractional Diffusion Equation with Implicit Difference Method / Chunye Gong, Weimin Bao, Guojian Tang, Yuewen Jiang, Jie Liu // The Scientific World J. Vol. 2014. Art. ID 219580. 8 p.

4. Parallel Algorithm with Parameters Based on Alternating Direction for Solving Banded Linear Systems / Xinrong Ma, Sanyang Liu, Manyu Xiao, Gongnan Xie // Math. Probl. in Engin. Vol. 2014. Art. ID 752651. 8 p.

5. Amdahl G. M. Validity of the single processor approach to achieving large scale computing capabilities // Proc. AFIPS spring joint comp. conf. Reston, VA: AFIPS Press, 1967. P. 483–485.

6. Introduction to Parallel Computing, Second Edition / A. Grama, A. Gupta, G. Karypis, V. Kumar. USA: Addison Wesley, 2003.

7. Yang T., Gerasoulis A. DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors // IEEE Trans. Parallel and Distributed Systems. 1994. Vol. 5, № 9. P. 951–967.

8. Liu C. L., Layland J. W. Scheduling Algorithms for Multiprogramming in Hard Real-Time Environment // J. ACM. 1973. Vol. 20, № 1. P. 46–61.

9. Kupriyanov M. S., Shichkina Yu. A. Applying the list method to the transformation of parallel algorithms into account temporal characteristics of operations // Proc. of the 19th Intern. Conf. on Soft Computing and Measurements. SCM 2016, 7519759. P. 292–295. ISBN: 978-146738919-8, DOI: 10.1109/SCM.2016.7519759.

10. Kupriyanov M. S., Shichkina J. A., Al-Mardi M. Optimization algorithm for an information graph for an amount of communications // Lecture Notes in Comp. Scie. (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Vol. 9870, 2016. P. 50–62. ISSN: 03029743 ISBN: 978-331946300-1, DOI: 10.1007/978-3-319-46301-8_5.

Yu. A. Shichkina, M. H. A. Al-Mardi, M. S. Kupriyanov
Saint Petersburg Electrotechnical University «LETI»

OPTIMIZATION OF THE PARALLEL ALGORITHM BY REDUCING THE TIME FOR INTERPROCESSOR DATA EXCHANGE

In the article results of researches in the field of construction of parallel algorithms effective for time and volume of computing resources for distributed memory computing systems are presented. A method that allows to improve the schedule of parallel algorithm execution time at the expense of redistribution of operations between processes. As a result, the amount of transmitted messages between processes decreases and the time spent on interprocessor communication is reduced. In implementing the method of volume of computing resources does not change. The method is based on the adjacency list, corresponding to the information graph of the algorithm. The method can be applied in combination with other methods for constructing a schedule for the execution of algorithms, for example, oriented to optimizing the number of computational nodes, to achieve an optimal schedule for a number of parameters.

Parallel algorithm, algorithm optimization, information graph, operation execution time, algorithm execution schedule, process, processor, interprocessor data transmission